Fog and Edge Oriented Embedded Enterprise Systems Patterns: Towards Distributed Enterprise Systems That Run on Edge and Fog Nodes

Hamish Sadler
Queensland University of Technology
Hamish.Sadler@qut.edu.au

Alistair Barros
Queensland University of Technology
Alistair.Barros@qut.edu.au

Wayne Kelly
Queensland University of Technology
W.Kelly@qut.edu.au

Abstract

Enterprise software systems enable enterprises to enhance business and management reporting tasks in enterprise settings. Internet of Things (IoT) focuses on making interactions possible between a number of network-connected physical devices. Prominence of IoT sensors and multiple business drivers have created a contemporary need for enterprise software systems to interact with IoT devices. Business process implementations, business logic and microservices have traditionally been centralized in enterprise systems. Constraints like privacy, latency, bandwidth, connectivity and security have posed a new set of architectural challenges that can be resolved by designing enterprise systems differently so that parts of business logic and processes can run on fog and edge devices to improve privacy, minimize communication bandwidth and promote low-latency business process execution. This paper aims to propose a set of patterns for the expansion of previously-centralized enterprise systems to the edge of the network. Patterns are supported by a case study for contextualization and analysis.

1. Introduction

Enterprise systems facilitate the integration and extending of business processes and workflows among different business functions within an organization and between multiple organizations [1]. With focus on satisfying the needs of an enterprise, enterprise software systems cover a wide range of functionality like project management, online payments, business process management and enterprise content management [2].

The Internet of Things (IoT) focuses on making interactions between a number of network-connected physical devices possible via wireless data communication over the Internet as a global communication medium [3]. Social graph analysis and management, big data, and cloud data management, ontological modeling, smart devices, personal information systems, and non-functional requirements such as location-independent response times, security and privacy are some of the typical challenges and concerns of IoT architectures [4]. Security and privacy in IoT with concerns like object identification, authentication and authorization, software vulnerabilities and backdoor analysis as well as malware remain some of the most important issues to address [5].

Edge computing is a computing paradigm in which data is processed at the edge of the network where most data is generated [6]. As opposed to cloud computing, this paradigm helps address issues and limitations like time sensitivity and data volume [7].

Low latency requirements, network bandwidth constraints, resource-constrained devices and uninterrupted services with intermittent connections have all been cited as some of the architectural challenges that have led to the concept of "fog computing". In fog computing, rather than relying upon centralized cloud or the processing power of edge devices, computation, communication, control and storage responsibilities are handled by fog nodes that are closer to the edge of the network [8]. Fog computing has the characteristics of low latency, location awareness, geographic distribution, end device mobility, capacity of processing high number of nodes, wireless access, real-time applications and heterogeneity [9]. Fog computing has been cited to be useful for healthcare, urgent computing, smart energy grids, vehicular fog computing and Vehicular Ad-hoc Networks(VANET), smart environments, augmented reality, brain machine interface and gaming [10]. One important aspect with fog computing is the fact that with many heterogenous devices joining and leaving fog networks openly, they create huge and high-frequency volumes of data [11]. This aspect brings up a range of interesting challenges when integrating IoT in enterprise systems on a fog network.

In a traditional sense, the interactions of enterprise systems with IoT devices have been treated mostly under "integration of IoT with enterprise systems". In this paradigm, enterprise systems mostly reside centrally on-premises or in the cloud and somehow interact with IoT devices using some middleware or IoT gateway. Some of the same architectural concerns like low latency, high data volumes and frequency, security and privacy that drove the concept of fog computing can also be drivers to take a different approach to designing enterprise systems that are heavily integrated with IoT devices. Most importantly, in this paradigm, business logic and microservices can be pushed to fog and edge nodes which helps address some of these common concerns. This paradigm leads to the need for proposing a set of architectural patterns focused on pushing enterprise systems all the way to the edge of the network under the concept of "embedded enterprise systems", a field left partially unattended in the literature.

Aligned with the move towards distributed embedded enterprise systems, this paper tries to provide a non-exhaustive list of architectural design patterns that could help systemize common solutions to common problems faced during this architectural shift. It also tries to articulate the trade-offs between the gains and the losses of such transition.

This paper includes six sections. Section 2 describes the architectural paradigm shift from IoT-integrated enterprise systems towards embedded distributed enterprise systems around a case study. Section 3 discusses previous work. Section 4 presents a list of patterns for the design of embedded enterprise systems. Section 5 presents a discussion on the design trade-offs between centralized and distributed embedded enterprise systems and finally, Section 6 includes a conclusion.

2. Research Domain and Method

Conventional development paradigms of enterprise systems have traditionally relied upon some underlying technologies and concepts like Business Process Management (BPM), Information Integration and Interoperability, Enterprise Architecture and Enterprise Application Integration and Service-Oriented Architecture (SOA) [1]. However, Internet of Things enabled sensors and internet-connected devices at masses have created an environment where a huge volume of data can be collected from the environments in which such enterprise systems operate. In an era of high-frequency and private big data where the volume of the data generated, collected and sensed by various devices is more than the amount that can be processed, for enterprise systems to evolve into their next generation, their design has to also evolve. Examples include hospital patient management systems that can receive live patient status monitoring data directly from patient-connected sensors, school/student management systems that receive attendance and student entry/exit notifications from IoT sensors or mobile devices, power grid management systems that receive status updates from IoT sensors, enterprise procurement systems that receive live updates from IoT sensors alongside the delivery or procurement pipeline or manufacturing management systems that receive live updates from a range of sensors for purposes like efficiency and predictive repair management. While residing in the cloud or on-premises, a traditional IoT-integrated enterprise system may interact with IoT devices through an IoT gateway and an edge tier, as illustrated in **Figure 1**. However, a range of architectural concerns like high volumes of data, challenges with transferring such volumes of data to the central enterprise systems combined with the needs to filter, refine and find patterns in the sensed data as well as tight privacy requirements require a new way of thinking in designing future-focused enterprise systems where at least parts of the core business logic can run closer to the edge of the network. This evolution of architecture has been illustrated in **Figure 1**.

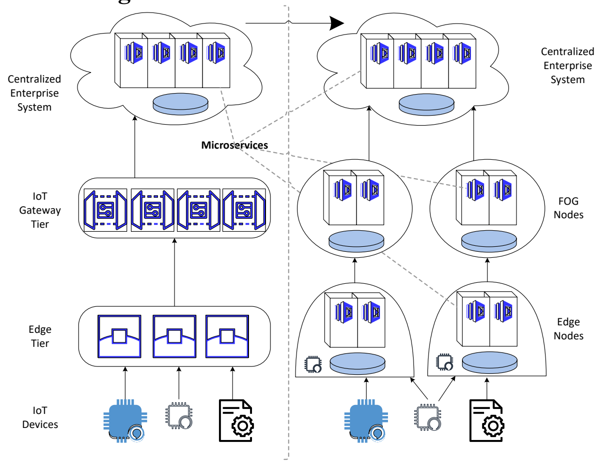

Figure 1 Evolution from IoT-integrated enterprise systems (left) to distributed embedded enterprise systems that run on fog and edge nodes (right)

In contrast to typical cloud-based designs, this new style of building embedded enterprise systems is driven by a range of architectural concerns. Such concerns include security and privacy covering the storage of data on edge and fog nodes as well as the communication of data from such edge and fog nodes back to the core enterprise system. Private or sensitive data may be collected and handled and communicated by such nodes and without provisions in place, this can lead to unauthorized disclosure of such data or security breaches. Some of the basic security and privacy constructure like authentication, data integrity, non-repudiation and authorization have to be devised and thought of before being able to break down an enterprise system to run on edge and fog nodes. There are also concerns that unlike cloud-hosted resources, fog nodes may not be maintained under tight-enough physical security protocols [10, 12, 13] making this evolution even further challenging. Performance is another architecture concern that has to be taken into account. This covers aspects like limited computing capacity, limited battery capacity, limited communication bandwidth as well as limited memory [13-15]. Device Management is another aspect that can become a logistical challenge when running and orchestrating execution of enterprise systems on edge and fog nodes. This aspect covers areas like device discovery, deployment, execution orchestration, remote access and management, device naming and identification [10, 16]. High data volumes and communication bandwidth constraints stand out as architectural drivers to evolve typical enterprise systems in advanced manufacturing [17, 18], as well as predictive maintenance and line downtime minimization via Cyber–Physical Systems and digital twins which form the foundations of smart manufacturing together with IoT [19].

One important aspect in IoT-enabled enterprise systems is the abstraction levels applicable to data, from sensors that generate raw data, to digital twins and at a higher level, to a business object, which is relevant to enterprise systems. Refinement of events, filtering them and applying business rules to the sensed data are big differentiators necessary to consider when using IoT-generated data within enterprise systems. For example, it is sensible for a blood pressure sensor connected to a patient for a patient management system to be aware of the business rules and conditions around that patient and accordingly, trigger the execution of certain business processes and logic without necessarily having to communicate every bit of data back to the central patient management and monitoring system. In this sense, the correlation between physical assets, their digital twins and their presence in enterprise systems are not easy to understand, leading to the creation of needs to re-think the use of cloud computing, reshaping thinking around running parts of enterprise systems on the edge of the network or at least close to where the data is generated rather than the cloud.

Accordingly, **Table 1** lists a set of architectural concerns that drive the evolution of conventional IoT-integrated enterprise systems into distributed IoT-enabled embedded enterprise systems.

Table 1 Architectural concerns driving the evolution of IoT-integrated enterprise systems into distributed IoT-enabled embedded enterprise systems

Concern Title	Description
Privacy	*Consumer privacy requirements may require the storage of private data on the edge of network in user-trusted zones [9]*
Security	*Maintaining data closer to user-owned edge devices can improve security [20]*
Bandwidth	*Bandwidth limitations may prohibit easy communications to centralized cloud systems [21]*
Reliability	*High volume data requires a more localized and regionalized setup for connectivity [20, 22]*
Low latency and real-time communication	*Local processing of high volumes of data generated by IoT sensors requires low-latency real-time communications [20]*
Mobility	*Edge devices may be mobile and may join and leave localized fog networks [20]*

These requirements in the design of IoT-integrated enterprise systems can lead to a set of common design and architecture themes for which there could be common solutions as patterns. Such patterns can help enterprise and solutions architects design such new generations of enterprise systems with more insights. This paper attempts to provide a set of such common architectural themes and provide a set of common solutions for such themes in an attempt to make the design of such embedded enterprise systems more streamlined and straightforward for its architects. And in doing so, it contextualizes a manifesto around a revelatory case study as the research method.

2.1. Case Study: Xiippy.ai, Privacy Preservation in the Context of Enterprise Customer Relationship and Loyalty Management

A Customer Relationship Management (CRM) system is a system used by organizations to manage their interactions and relationships with their customers using data analysis over large datasets [23]. A Customer Loyalty Management System is a system used by organizations to design, create, manage, and analyze loyalty programs [24]. There is empirical evidence of direct correlations between the effectiveness of marketing and loyalty management and utilizing customers' purchase history [25]. Customer relationship management can be heavily affected by regulatory and compliance related aspects due to requirements around personal data collection. One key driver in this space is that personalization opportunities increase significantly for registered customers [26]. These observations point towards an organization's tendency for collecting customer details to build a database of customers [27]. However, such needs are heavily under the impact of regulatory limitations. For example, California's Song-Beverly Credit Card Act, Civil Code section 1747.08 prohibits offline retailers from requesting or requiring "personal identification information" (PII) in connection with consumer credit card transactions [28]. This shows that the goal of building a database of customers for offline retailers can be quite difficult if not impossible. Most traditional approaches on designing an enterprise Customer Relationship Management and Customer Loyalty Management system are based upon the assumptions that such enterprises own their customers' details and even their purchase history. As a result, for physical retail enterprises, building effective CRMs and loyalty management systems are under the influence of certain "requirements" around consumer privacy and data ownership that point towards having to take a different approach on designing such systems.

We hereby propose a contemporary platform that has taken a different approach on building an enterprise CRM and loyalty management system for the retail industry where collecting personal details at the counter and owning such data may not be feasible, especially for all customers.

Xiippy.ai is a multi-patented privacy-preserving data-rich payments, receipts, loyalty, rewards and customer relationship management platform for retail. Xiippy's web-based enterprise dashboard is used by retail enterprises to manage concepts and constructs like customer relationship, customer engagement, customer loyalty, orders, reports, campaigns and access/permission/roles control and is an example of an enterprise system that has been designed in a specific way to address some underlying requirements around consumer privacy and data ownership [29].

Retailers' avoidance to share itemized sales data, consumer privacy and lack of standardization have been cited to be the most prominent barriers against the adoption of digital receipts [30]. The underlying logic prohibiting outcomes in this space is that no party can be trusted by all retailers and shoppers at the same time to be the source of truth for all sales and purchase history and that consumers are not comfortable providing personal details at the counter to identify themselves. These are some of the issues Xiippy.ai resolves [29].

For Xiippy, privacy needs of retail enterprises and their customers have been a strong driver to take a different design approach for enterprise customer relationship and loyalty management systems, especially aimed for retail franchise enterprises, whereby business logic that determines rewards eligibility is pushed all the way to the edge of the network, on consumers' personal devices where private purchase history is securely stored. To protect privacy, end-to-end encryption is used to transfer digital receipts and statements directly from Point-of-Sale systems, operated by merchants to consumers' personal devices. This helps establish a private channel to transfer itemized statements while the data transfer intermediary (i.e. Xiippy.ai) maintains no knowledge of such data. Merchants' copy of statements together with privacy-preserving customer identifiers are made available to merchants under the enterprise dashboard, as illustrated in **Figure 2** [31].

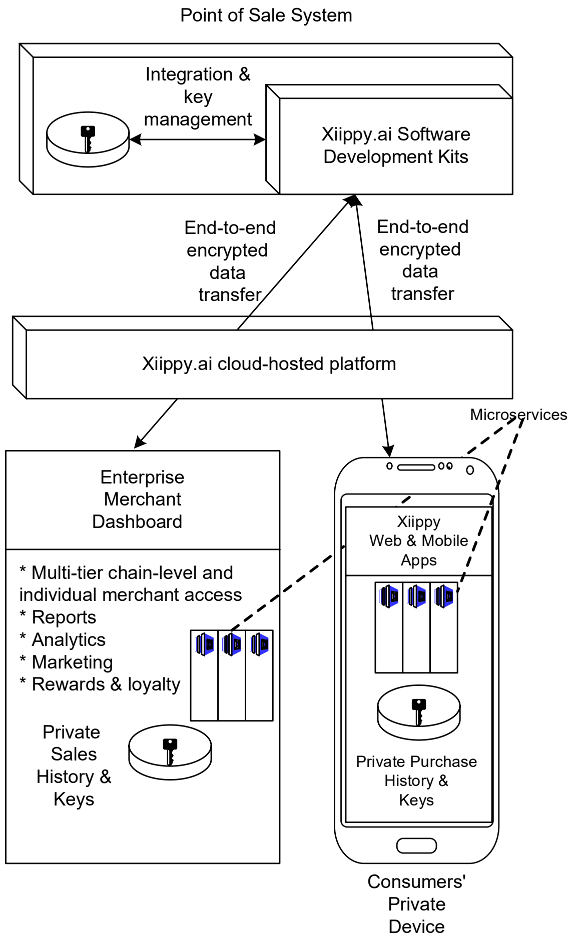
Point of Sale System
Integration & key management
Xiippy.ai Software Development Kits
End-to-end encrypted data transfer
End-to-end encrypted data transfer
Xiippy.ai cloud-hosted platform
Microservices
Enterprise Merchant Dashboard
* Multi-tier chain-level and individual merchant access
* Reports
* Analytics
* Marketing
* Rewards & loyalty
Private Sales History & Keys
Xiippy Web & Mobile Apps
Private Purchase History & Keys
Consumers' Private Device

Figure 2 The high-level architecture of the Xiippy.ai platform [31]

At the core of the Xiippy platform, sits its Point-Of-Sale (POS) Software Development Kits (SDKs) that receive transaction data from the POS and facilitate an end-to-end encrypted transfer of digital statements to consumer devices. It also enables an end-to-end encrypted transfer of merchants' copy of such statements to the merchants' enterprise dashboard. Xiippy uses the keychain storage of consumer devices as well as WebCrypto APIs to securely store private cryptographic keys on the edge of the network that facilitate the decryption of data at the edge [31].

With private purchase history stored on private consumer devices, business processes related to rewards eligibility detection can be executed on such devices. This an example of executing microservices on the edge of the network that in other design paradigms are traditionally part of central enterprise systems.

3. Previous Work

Previous attempts to standardize patterns in systems and software design include those of E. Gamma et al. providing 23 solutions to commonly occurring problems in object oriented design [32]. An example of a set of software design principles is R. Martin's object-oriented design principles, commonly known as the SOILD principles [33] guiding systems developers towards creating more maintainable and loosely-coupled code. Other attempts include service-oriented design patterns T. Erl [34, 35] and cloud-specific design patterns [36-38], providing a set of service-oriented design rules to guide the development of service-oriented solutions. Attempts to catalog enterprise integration patterns include those of G Hohpe et al. with a great deal of focus on building messaging mechanisms around enterprise systems [39]. Barnes' set of architecture evolution styles is another example of focusing on abstraction and componentization of code blocks using formal language structures [40]. Another example of using a formal language model for describing evolution patterns is [41] with examples like edge provisioning, edge code deployment, edge orchestration and edge diameter of things (DOT) patterns which mostly cover IoT and edge-only scenarios without covering enterprise systems integration aspects. There are also those who have tried to articulate on-premises to cloud migration patterns. Examples include [42] with patterns like Re-deployment, Cloudification, Relocation and Multi-Cloud Refactoring patterns which mostly include non-IoT and cloud-only scenarios without covering enterprise systems integration with IoT for high-volume high-privacy settings.

In IoT, a range of previous attempts have been put into compiling architecture patterns. Hasan Derhamy et. al's list of commercial IoT frameworks is an example which have been defined as "a set of guiding principles, protocols and standards which enable the implementation of Internet of Things applications", with examples including frameworks like IPSO Alliance, IoTivity, AllJoyn and Thread [43]. R Tkaczyk et al.'s catalogue of design patterns for IoT artefact integration include 13 IoT artefact integration patterns which mostly cover non-fog settings without digging deep into the problem of IoT-integrated enterprise systems [44]. Another example includes [45] where generic software design patterns have been contextualized in an IoT setting. A set of IoT interoperability-focused design patterns have also been catalogued by Rafał Tkaczyk et al. in [46] which includes examples like IoT Gateway Event Subscription and D2D REST Request/Response. Privacy and security specific challenges in IoT have been discussed in [9, 47] that include trust, privacy preservation, authentication and key agreement, intrusion detection, dynamic join and leave of fog node and cross-border issue and fog forensic. Security and privacy preservation patterns in fog computing include those of [48] with examples like Certificate-Less Aggregate SignCryption scheme (CLASC), aimed for vehicular crowdsensing using fog computing. Another example is Patrik Spiess et. Al's proposal around an SOA-based architecture for the integration of IoT in enterprise systems which they name SOCRADES that helps hide the heterogeneity of hardware, software, data formats and communication protocols of IoT and embedded systems [49] leaving fog-oriented scenarios uncovered. Stephan Haller et al. propose the concept of decomposed business processes at three layers of backend, network and edge devices to enable localized distributed decision making [50] as another example of attempting to integrate IoT within enterprise systems which also leaves fog-networking scenarios uncovered. Matthias Thoma et al. attempt to merge ideas from the Internet of Services (IoS) and the enterprise IT world for describing and provisioning "IoT-services" [51] forming another example of merging IoT in enterprise systems. Alfred Zimmermann et al. propose a metamodel-based approach for integrating Internet of Things architectural objects [52]. P. Fremantle's reference architecture for IoT provides a layered structure for the internet of things which includes the layers of client/external communications, event processing and analytics, aggregation/bus, relevant transports , and devices [53] which also leaves architectural concerns relevant to high-data-traffic high-privacy IoT integrated enterprise systems uncovered. D. Repta et al.'s efforts in formulating the concept of Cyber Intelligent Enterprise is yet another attempt to propose a way forward for integrating physical objects in virtual environments which covers the three main goals of information processing, domain representation and object abstraction [54].

A review of the previous work in integrating fog-connected edge devices within enterprise systems has revealed that the problem of evolving the architecture of a large-scale and potentially monolithic enterprise systems to an open distributed IoT-enabled setting has not been fully addressed or analyzed via common themes or problems or styles or patterns in the literature, leaving the field of fog and edge oriented embedded enterprise systems unaddressed.

4. Fog and Edge Oriented Enterprise Systems Patterns

This chapter proposes a non-exhaustive set of architectural design patterns for edge and fog oriented distributed embedded enterprise systems with each pattern highlighting a problem, a solution and some use cases for contextualization.

4.1 Intermediated Eventing Pattern

Problem

Enterprise systems that require receiving events and data from IoT devices, mostly require events at a higher abstraction layer and lower frequency. For such enterprise systems, anomalies in input data streams that belong to business objects have a much higher level of significance compared to raw data streams. How might we filter, analyze, detect and transform raw data into meaningful data that such enterprise systems expect?

Solution

The process to analyze raw data, apply sensible business logic to the data, derive and extract patterns and trigger a business process can happen on either edge nodes or fog nodes as a data flow intermediation layer. Provided that there is an initialization of state and business rules on fog and edge nodes, such nodes will possess all they need to detect anomalies or patterns in IoT-generated data streams and eliminate the challenge of having to communicate all the sensed data to the central enterprise system. The benefit of this pattern is to eliminate avoid high data volume conversations with the centralized system.

Use Cases

In geo-tracking farm cattle use cases, proximity to certain regions of the farm may be of significance rather than all locations the cattle traverse on the farm. The Intermediated Eventing Pattern can help transform and translate input data streams to of-interest output events. Another example includes IoT-enabled patient management systems in which only certain patterns in the sensed data need to be reported back to the central enterprise system.

4.2 Buffered Eventing Pattern

Problem

Intermittent network connectivity issues, high data volumes and high local reliability requirements for IoT devices that act as sources of data for enterprise systems make it difficult to send all the data to such systems. How might we ensure no such data are lost despite the high volumes?

Solution

Rather than trying to send all the sensed data directly to the enterprise system, edge IoT devices and sensors can rely upon fog nodes to act as buffers to IoT data and events before publishing them to cloud-hosted brokers. This enables local resiliency and helps resolve the intermittency of connectivity to central cloud. The benefit of this pattern is to improve the reliability of the enterprise system and ensure no enterprise-important data is lost without the need to send all data to the centralized system in real-time.

Use Cases

In real-time health analytics and monitoring, the Buffered Eventing Pattern can help deal with high data volumes, intermittent connectivity and availability issues hence preventing chances of critical data loss.

4.3 Business Object to Thing Multi-Binding Pattern

Problem

Enterprise systems own their users and trust models. A mapping of enterprise business objects to IoT devices needs to be securely established before data and events can be deemed to belong to such enterprise business objects. This issue gets more complex when the same IoT device is supposed to interact with more than one enterprise system. How might we make it possible for an IoT device to interact with multiple enterprise systems interested in receiving data and events from such devices?

Solution

Through an initial mapping exercise, the identity of the enterprise business object is diffused and extended to the IoT device(s), owned by or relevant to such business objects. This mapping or binding exercise could include the issuance of temporary access tokens digitally signed by an identity provider trusted by the enterprise system. Any further interactions of edge devices with fog nodes, cloud components or the enterprise system itself can be validated by validating the digital signatures of the supplied tokens by the edge device. Locally-deployed microservices and business logic must handle state management and maintenance of such tokens in cases where multiple tokens from multiple enterprise systems are to be used to interact with multiple enterprise systems. The benefit of this pattern is to facilitate “thing sharing” between multiple independent enterprise systems while maintaining a robust and secure trust model.

Use Cases

In IoT enabled procurement and order fulfillment enterprise systems, the same IoT sensor may need to interact with both procurement and order fulfillment systems. Examples include drone-delivered goods or goods that must remain under constant monitoring during shipping. In such settings, a multi-binding between the same IoT device and multiple enterprise systems can create the fluidity and dynamicity required.

4.4 Hot Microservice Swapping Pattern

Problem

In settings where edge devices join commodity fog networks, there may be a need for loading and executing business process microservices specific to that fog network or specific to a regional enterprise system. How might we enable the seamless execution of business logic from such enterprise systems despite the fluidity of the commodity fog network?

Solution

As an initialization step, upon joining a commodity network, the IoT device must establish trust and receive state, business rules and microservice logic from the relevant enterprise system to be able to process local data and events and apply relevant business rules to them. The benefit of this pattern is to enable centralized enterprise systems to interact and integrate with a dynamic list of IoT devices on the fly.

Use Case

Downloading region-specific business process microservices on smart cars that would dynamically extend a regional traffic management system all the way to the edge of the network requires the hot swapping of such microservices as the car can fluidly join and leave such commodity local fog networks. A regional traffic management system can then enforce different business rules to the sensed data.

4.5 Democratized Trust Pattern

Problem

In multi-layered fog networks where it is a requirement for the data and events created by edge devices to be persisted as they arrive irrespective of connectivity between cloud-to-fog and fog-to-fog nodes, the refresh of temporary credentials issued to edge devices may need to happen when connectivity to the centralized cloud identity provider is not available. How might we make it possible to continue authenticating edge devices fulfilling the reliability of data/event collection without having to refresh credentials via a trusted centralized identity provider?

Solution

Certain building blocks like identity providers can be replicated at the middle fog layers to provide autonomy in authentication and authorization after the initial authentication and authorization of edge devices using a centralized identity provider. Via an initial establishment of trust between all identity blocks at the fog levels, each of these middle-layer identity providers can remain enabled to issue new temporary credentials. The benefit of this pattern is to minimize dependencies on centralized access, authentication and authorization control in settings with intermittent connectivity issues.

Use Cases

In healthcare settings where edge devices report patients' details to fog nodes that are locally hosted at the hospital level as well as to a patient management system, the process to keep recording the generated data while relying upon temporary credentials can be achieved using the Democratized Trust Pattern.

4.6 Protected Data Synchronization Pattern

Problem

Given the privacy and security needs in IoT and assuming that the only trusted zone for plain data to exist in is the edge of the network, limited storage capacity makes it difficult to store large volumes of data possible. This alludes to the need to rely upon cloud storage or fog node storage as the long-term permanent storage of data. How might we preserve privacy while treating public cloud or fog nodes as the long-term storage provider for private IoT data?

Solution

Rather than maintaining large volumes of data on the edge, only the portion of data that is required is retrieved and processed on the edge in plain format. Fog nodes and cloud nodes are then used to only maintain encrypted data that have been encrypted with keys only maintained on the edge devices. This facilitates synchronization mechanisms in which cloud and fog nodes only maintain non-plain data. The benefit of this pattern is to make cloud storage for inherently-private data possible to extend the limited storage capacity of edge and IoT devices privately and securely.

Use Cases

For privacy-preserving big data and federated machine learning settings like purchase needs prediction, the Artificial Intelligence (AI) models can be trained on edge devices. In settings where multiple devices belong to the same user, the Protected Data Synchronization Pattern can be used to synchronize private datasets on multiple edge devices provided all such devices can establish a common trust model with their owner. Edge-executed microservices can then be used to determine predictive offers exposed by enterprise systems hence eliminating the need for such enterprise systems to maintain knowledge of consumers' purchase history.

4.7 Zero-Knowledge Data Transfer Intermediacy Pattern

Problem

An enterprise system may need to receive data and events from edge devices that are connected to commodity fog networks. Such commodity fog networks must maintain zero knowledge of the contents of such data despite transferring data to the enterprise system. How might we enable the persistence of edge-generated data securely on commodity fog nodes?

Solution

Given that the enterprise system and edge devices are deemed the only trusted zones where data can exist in plain format, following and initial establishment of trust, such edge devices and the enterprise system utilize a public key infrastructure to exchange public keys and agree upon common secrets to encrypt and decrypt data while utilizing the fog network as a data transfer medium. The benefit of this pattern is to facilitate private secure communication between an enterprise system and commodity devices that dynamically join a network and require private secure communication with the enterprise system.

Use Cases

High-volume sensor data in vehicular fog computing settings that are to be shared with traffic or smart city enterprise systems may need to be persisted on fog nodes first to be later on transferred to the right recipients. In such settings, the Zero-Knowledge Data Transfer Intermediacy Pattern could be used to eliminate the need for real-time transfer of high data volumes to the central enterprise system.

4.8 Distributed State Synchronization Pattern

Problem

Given that the IoT edge nodes or fog nodes can run microservices and may require to maintain enterprise-important state to apply business rules and logic to the sensed data locally, it is important for the central enterprise system to synchronize state with edge and fog nodes to maintain integrity of state and apply the relevant business processes and rules to such state at a global level. How might we keep the integrity of global state while allowing the distribution of state?

Solution

Regional fog nodes must be initialized with a set of business rules to achieve a locally-wholistic view of state and instruct edge devices with the right updated business rules to maintain integrity of operations. They can then synchronize state with the central enterprise system lazily. The benefit of this pattern is to make it possible to orchestrate a distributed but common state management model.

Use Cases

In a healthcare setting where IoT sensors report the status of a patient to an enterprise patient management system, multiple independent sensors monitoring the same patient must synchronize their local state and business rules to execute a wholistically-correct set of steps against the same patient when it comes to triggering alarms and detecting anomalies in the sensed data. The Distributed State Synchronization Pattern can help in such scenarios to ensure a wholistically-correct but yet distributed state management mechanism.

5. Discussion

The decision to distribute enterprise systems and expand them to the edge of the network comes with a range of trade-offs between gains and losses. Accordingly, **Table 2** lists a number of such trade-off points.

Table 2 Architectural trade-offs between a traditional centralized design of IoT-integrated enterprise systems and distributed embedded enterprise systems

Trade-off Aspect	Description
Privacy	*In cases where privacy requirements make centralized storage of data difficult, a distributed embedded enterprise system design can have advantages.*
State Management	*State management in centralized systems is simpler to handle.*
Event Orchestration, clock and time synchronization	*Centralized orchestration of business process steps and events is much simpler in centralized systems.*
Reliability	*A distributed embedded design of enterprise systems can provide a higher level of reliability & fault tolerance due to minimum dependencies on centralized systems.*
Low latency and real-time responsiveness	*A distributed embedded design of enterprise systems fulfills latency and real-time response requirements better.*
Decision Making Locality	*Decision making on business process execution and triggering can be localized in a distributed embedded enterprise system in a more effective way.*
Complexity	*The design of a distributed embedded enterprise system imposes a range of complexities that are not normally faced with centralized systems.*
Scalability	*A distributed embedded design for enterprise systems provides a higher degree of scalability due to minimizing dependencies on centralized resources.*

Security	*Centralized enterprise systems are simpler to secure whereas distributed enterprise systems impose a range of security issues as a result of being physically distributed among many edge or fog nodes*
Operations Management	*Deployment, distribution, release management and disaster recovery in distributed embedded enterprise systems is inherently harder than centralized ones due to distributed state management and storage.*

6. Conclusion

Enterprise software systems are used in enterprise settings for a range of purposes mostly covering the integration and execution of business processes. In an IoT-enabled world, the need for the interaction of enterprise systems to IoT devices have increased significantly, however, aspects like high data volumes, privacy, reliability, and low latency requirements have made it challenging to simply integrate centralized enterprise systems with IoT devices. Fog computing and its benefits in terms of inclusion of localized close-by computing and storage capacity can solve certain problems in IoT-enabled enterprise systems. Based upon a case study for privacy-preserving customer loyalty and relationship management, this paper presents a non-exhaustive catalogue of patterns around a paradigm in which enterprise systems are designed to be distributed inherently and expand into fog and edge nodes with possibilities for the execution of microservices on edge and fog nodes as an enabler to resolve some of these specific requirements. As a limitation, this research does not include an exhaustive list of applicable patterns for edge and fog enabled embedded enterprise systems and further work can lead to a wider range of such patterns.

7. References

[1] L. Da Xu, "Enterprise systems: state-of-the-art and future trends," *IEEE Transactions on Industrial Informatics,* vol. 7, no. 4, pp. 630-640, 2011.

[2] R. Giachetti, *Design of enterprise systems: Theory, architecture, and methods*. CRC Press, 2016.

[3] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," *Computer Communications,* vol. 54, pp. 1-31, 2014.

[4] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*. Addison-Wesley Professional, 2003.

[5] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities," in *2014 IEEE 7th international conference on service-oriented computing and applications*, 2014: IEEE, pp. 230-234.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal,* vol. 3, no. 5, pp. 637-646, 2016.

[7] redhat.com. "Cloud vs. edge." https://www.redhat.com/en/topics/cloud-computing/cloud-vs-edge (accessed.

[8] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal,* vol. 3, no. 6, pp. 854-864, 2016.

[9] M. Mukherjee *et al.*, "Security and privacy in fog computing: Challenges," *IEEE Access,* vol. 5, pp. 19293-19304, 2017.

[10] C. Zhou and X. Zhang, "Toward the Internet of Things application and management: A practical approach," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014: IEEE, pp. 1-6.

[11] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*: Springer, 2014, pp. 169-186.

[12] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability," in *2015 IEEE International Conference on Mobile Services*, 2015: IEEE, pp. 313-319.

[13] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics,* vol. 10, no. 4, pp. 2233-2243, 2014.

[14] H. Yin, Y. Jiang, C. Lin, Y. Luo, and Y. Liu, "Big data: transforming the design philosophy of future internet," *IEEE network,* vol. 28, no. 4, pp. 14-19, 2014.

[15] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a smart city internet of things platform with microservice architecture," in *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015: IEEE, pp. 25-30.

[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials,* vol. 17, no. 4, pp. 2347-2376, 2015.

[17] D. Mourtzis, E. Vlachou, and N. Milas, "Industrial big data as a result of IoT adoption in manufacturing," *Procedia cirp,* vol. 55, pp. 290-295, 2016.

[18] J. Wan *et al.*, "Toward dynamic resources management for IoT-based manufacturing," *IEEE Communications Magazine,* vol. 56, no. 2, pp. 52-59, 2018.

[19] F. Tao, Q. Qi, L. Wang, and A. Nee, "Digital twins and cyber–physical systems toward smart manufacturing and industry 4.0: Correlation and comparison," *Engineering,* vol. 5, no. 4, pp. 653-661, 2019.

[20] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications,* vol. 98, pp. 27-42, 2017.

[21] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Communications Magazine,* vol. 56, no. 5, pp. 46-52, 2018.

[22] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37-42.

[23] M. Bardicchia, *Digital CRM-Strategies and Emerging Trends: Building Customer Relationship in the Digital Era*. 2020.

[24] g2.com. "Loyalty Management Software." https://www.g2.com/categories/loyalty-management (accessed.

[25] P. E. Rossi, R. E. McCulloch, and G. M. Allenby, "The value of purchase history data in target marketing," *Marketing Science,* vol. 15, no. 4, pp. 321-340, 1996.

[26] S. Legler. "Digital CRM 2.0 Building customer relationships in the digital landscape." https://www2.deloitte.com/content/dam/Deloitte/de/Documents/strategy/Deloitte_Digital_Digital_CRM_Study_2.0_2019.pdf (accessed.

[27] M. O. E. A. a. Communications, "Feasibility Study Showcase D: Borderless Real-Time Economy (RTE) Spearhead: eReceipt," 2019. [Online]. Available: https://likta.lv/wp-content/uploads/2019/08/eReceipt-Feasibility-Study_22.08.2019_final.pdf.

[28] casetext.com, "Capp v. Nordstrom, Inc.," 2013. [Online]. Available: https://casetext.com/case/capp-v-nordstrom.

[29] Xiippy.ai. "Xiippy.ai - Frequently-Asked Questions." https://xiippy.ai/FrequentlyAskedQuestions (accessed.

[30] K. F. Johannes Hübner, Fabian Schmid, Bedrija Hamza. "Digital Receipt Study: Drivers and Barriers to Adoption of Digital Receipts." https://www.autoidlabs.ch/projects/digital-receipts-for-the-consumer-iot/ (accessed.

[31] Xiippy.ai. "Xiippy.ai Security Architecture." https://xiippy.ai/security-architecture (accessed.

[32] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[33] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[34] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[35] T. Erl, *Soa: principles of service design*. Prentice Hall Upper Saddle River, 2008.

[36] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson, "Cloud Design Patterns," *Microsoft,* 2014.

[37] "Cloud Patterns." http://cloudpatterns.org (accessed.

[38] T. Erl, R. Puttini, and Z. Mahmood, *Cloud computing: concepts, technology & architecture*. Pearson Education, 2013.

[39] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[40] D. Garlan, J. M. Barnes, B. Schmerl, and O. Celiku, "Evolution styles: Foundations and tool support for software architecture evolution," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, 2009: IEEE, pp. 131-140.

[41] D. Tamzalit and T. Mens, "Evolution patterns: Designing and reusing architectural evolution knowledge to introduce architectural styles," *arXiv preprint arXiv:1605.06289,* 2016.

[42] P. Jamshidi, C. Pahl, S. Chinenyeze, and X. Liu, "Cloud migration patterns: a multi-cloud service architecture perspective," in *Service-Oriented Computing-ICSOC 2014 Workshops*, 2015: Springer, pp. 6-19.

[43] H. Derhamy, J. Eliasson, J. Delsing, and P. Priller, "A survey of commercial frameworks for the internet of things," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2015: IEEE, pp. 1-8.

[44] R. Tkaczyk *et al.*, "Cataloging design patterns for internet of things artifact integration," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018: IEEE, pp. 1-6.

[45] M. Koster, "design patterns for an internet of things—a design pattern framework for IoT architecture," *Iot-datamodels. blogspot. com,* 2014.

[46] S. Qanbari *et al.*, "IoT design patterns: computational constructs to design, build and engineer edge applications," in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016: IEEE, pp. 277-282.

[47] W. Wei, A. T. Yang, W. Shi, and K. Sha, "Security in internet of things: Opportunities and challenges," in *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, 2016: IEEE, pp. 512-518.

[48] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal,* vol. 4, no. 3, pp. 772-782, 2017.

[49] P. Spiess *et al.*, "SOA-based integration of the internet of things in enterprise services," in *2009 IEEE international conference on web services*, 2009: IEEE, pp. 968-975.

[50] S. Haller, S. Karnouskos, and C. Schroth, "The internet of things in an enterprise context," in *Future Internet Symposium*, 2008: Springer, pp. 14-28.

[51] M. Thoma, S. Meyer, K. Sperner, S. Meissner, and T. Braun, "On iot-services: Survey, classification and enterprise integration," in *2012 IEEE International Conference on Green Computing and Communications*, 2012: IEEE, pp. 257-260.

[52] A. Zimmermann, R. Schmidt, K. Sandkuhl, M. Wißotzki, D. Jugel, and M. Möhring, "Digital enterprise architecture-transformation for the internet of things," in *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*, 2015: IEEE, pp. 130-138.

[53] P. Fremantle, "A reference architecture for the internet of things," *WSO2 White paper,* 2015.

[54] D. Repta, I. S. Sacala, M. A. Moisescu, and A. M. Stanescu, "Towards the development of a Cyber-Intelligent Enterprise System Architecture," *IFAC Proceedings Volumes,* vol. 47, no. 3, pp. 827-832, 2014.